Hamish Drogemuller

13515109

DAT602 Corvus Documentation and report

Contents:

Table of Contents

[1. Introduction: 2](#_Toc104463961)

[2. Game Description: 3](#_Toc104463962)

[2.1 Game Premise: 3](#_Toc104463963)

[2.2 Login: 3](#_Toc104463964)

[2.3 Home: 3](#_Toc104463965)

[2.4 Gameplay: 3](#_Toc104463966)

[2.5 Objective and Scoring: 3](#_Toc104463967)

[2.6 Administration: 3](#_Toc104463968)

[3. Storyboards: 4](#_Toc104463969)

[3.1 Login: 4](#_Toc104463970)

[3.2 Register: 4](#_Toc104463971)

[3.3 Home: 5](#_Toc104463972)

[3.4 Join Lobby: 5](#_Toc104463973)

[3.5 Create Lobby: 6](#_Toc104463974)

[3.6 Player List: 6](#_Toc104463975)

[3.7 Log out: 7](#_Toc104463976)

[3.8 Admin Window: 7](#_Toc104463977)

[3.9 GameBoard: 8](#_Toc104463978)

[3.10 Cannot Place/Move: 8](#_Toc104463979)

[3.11 Win Screen: 9](#_Toc104463980)

[3.12 Loss Screen: 9](#_Toc104463981)

[4. Screen design decisions: 10](#_Toc104463982)

[5. Entity Relationship Diagram: 11](#_Toc104463983)

[6. Crud Table: 12](#_Toc104463984)

[6.1 C.R.U.D Analysis: 13](#_Toc104463985)

[7. Procedures: 15](#_Toc104463986)

[7.1 Player Registration: 15](#_Toc104463987)

[7.2 Player Selection: 15](#_Toc104463988)

[7.3 Creation and Confirmation for a Game: 15](#_Toc104463989)

[7.4 Live Gameplay: 15](#_Toc104463990)

[7.5 Game Administration Functions: 16](#_Toc104463991)

[8. ACID 17](#_Toc104463992)

[Atomicity: 17](#_Toc104463993)

[Consistency: 17](#_Toc104463994)

[Isolation: 17](#_Toc104463995)

[Durability: 17](#_Toc104463996)

[9. Deviations from Milestone I Planning 18](#_Toc104463997)

[9.1 Crud Table: 18](#_Toc104463998)

[9.2 ERD Re-Design: 19](#_Toc104463999)

# 1. Introduction:

This document will serve as a planning report and documentation for the Corvus game database. The aim of this project is to create a tile based multi user game. This document aims to explain decisions behind design choices as well as plot the layout and follow the creation and evolution of the database.

**Milestone I**

# 2. Game Description:

## 2.1 Game Premise:

The game outlined in this document referred to as “Corvus” will revolve around the user trying to scare crows off of their area of the board. Points will be gained for how many crows you can get to land on the opponent’s field. A crow will take up 1 game tile and multiple will be on each field. The game will be played by two users at a time (one on each field) and the game will end when a player gets a 100 score or after 15 minutes of gameplay.

## 2.2 Login:

When the game is run, players will be prompted to log in with an email/username and password. If the email isn’t found in the database then the player will be asked if they wish to create an account. When registering an account the player will be required to enter a unique email and a password. However, if the email/username is already in the database and the password does not match what is stored against the email, the app will log the wrong attempts. The user will become locked out upon 5 wrong entries. Once an account is created the user will be prompted to log in again.

## 2.3 Home:

Upon player login to the game they will be provided a list of options; Quick join(Finds a random game), join lobby(allows the player to join from a list of available lobbies), create a game(allows the player to create a new lobby) and players(which provides a list of online players that could be invited to a lobby). Players will also have the option to log out. If the player has admin privileges, they will also be allowed to launch an administrator window.

## 2.4 Gameplay:

Upon entering a game the user will be spawned onto the gameboard, the gameboard consists of two tiled halves. The players half and the opponents half. Once both players have spawned in they will have 15 seconds to set up, this may include setting up traps or decoys for the birds or just positioning the character. After the 15 seconds is up the match will begin and the birds will spawn at random. Once the player, trap, or decoy is within an adjacent tile the birds will “fly” away. As stated above, the players will have access to two different item types. Decoys and traps, Decoys will deter birds from landing in an area around it while traps will startle birds in a larger area but will disappear after being used. Decoys will stay for a set amount of time, becoming less effective as the birds get used to it.

## 2.5 Objective and Scoring:

Scoring will be calculated based on how many birds are present on the opponent’s board, 1 bird counts for 1 point. The first player to 100 points wins. Scoring will be updated/calculated at 3-second intervals.

## 2.6 Administration:

Certain accounts will be given administrative privileges to assist in the running and maintaining of the Corvus player ecosystem, these obligations will range from assisting with locked accounts, forgotten passwords to deleting player accounts if they are abusive, cheating, etc.

# 3. Storyboards:

## 3.1 Login:

Graphical user interface

Description automatically generated

## 3.2 Register:

Graphical user interface, application

Description automatically generated

## 3.3 Home:

Graphical user interface, application, Word

Description automatically generated

## 3.4 Join Lobby:

A picture containing graphical user interface

Description automatically generated

## 3.5 Create Lobby:

Graphical user interface, application, Word

Description automatically generated

## 3.6 Player List:

A picture containing graphical user interface

Description automatically generated

## 3.7 Log out:

Graphical user interface, application, Word

Description automatically generated

## 3.8 Admin Window:

Graphical user interface, application, Word

Description automatically generated

## 3.9 GameBoard:

Scatter chart

Description automatically generated with medium confidence

## 3.10 Cannot Place/Move:

Graphical user interface, application, Word

Description automatically generated

## 3.11 Win Screen:

Graphical user interface, Word

Description automatically generated

## 3.12 Loss Screen:

Graphical user interface, application, Word

Description automatically generated

# 4. Screen design decisions:

**Overall:**

The overall design I wanted to encapsulate with the entire game application is a minimalistic, simple-to-follow design.

**Login:**

The login screen was designed to be minimal and easy to follow, Providing easy-to-follow steps that allow users to log in as well as register if it’s their first time on the app. Since the user has to sign up if their name/email isn’t in the system the login screen will also function as a registration screen. The user can put in an email and password, then be prompted for a username and any more information the app requires.

**Home:**

For the home screen, I wanted to maintain the minimal aesthetic but also provide a simple interface that doesn’t overwhelm the user with options, to this end I made sure that big groups of data were put behind headers. Such as hiding all of the available lobbies under the join lobby header. I also wanted this screen to be the “hub” for the game, allowing the user to do anything they need to do before getting into a game. This includes creating a game, joining a game, editing options, running the administration window if you have the privileges, and logging out of the game.

**Gameboard:**

I wanted the gameboard to feel like it was an open expanse that the user had to defend. To achieve this I decided to make the game space a 20x10 board. This would provide each player with a 10x5 space to defend, the size of each side means that the players can place traps and decoys but still have to move around to scarecrows.

**Admin:**

The admin screen is all about maintaining the game, to this extent I wanted to make it a concise screen with all the necessary tools ready to use upon screen boot. To this end, I included all available commands compactly on this screen.

# 5. Entity Relationship Diagram:

**Diagram

Description automatically generated**

# 6. Crud Table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Entity/Attribute | **Check For Username** | **Register Account** | **Check For Correct Password** | **Lock Account** | **Successful Login** | **Start New Game** | **Quick Join/Join Lobby** | **Player Moves** | **Scoring** | **Game Ends** | **Player Logs Off** | **Open Admin Console** | **Admin Kill Running Game** | **Admin Delete Player** | **Admin Creates New Player** | **Admin Edits Existing Player Info** | **Player sends a Chat Message** |
| **Player** | R | C |  |  | R | R |  | R | R |  |  | R |  | D | C | RU | R |
| Player ID |  | C |  |  |  | R |  | R | R |  |  | R |  | D | C | RU | R |
| Username | R | CR | R |  | R |  |  |  |  |  |  | R |  | D | CR | RU | R |
| Password |  | C | R |  |  |  |  |  |  |  |  |  |  | D | C | RU |  |
| Email |  | CR |  |  |  |  |  |  |  |  |  |  |  | D | CR | RU |  |
| Locked User |  | C |  | U |  |  |  |  |  |  |  |  |  | D | C | RU |  |
| Admin |  | C |  |  | R |  |  |  |  |  |  |  |  | D | C | RU |  |
| Login Attempts |  | C |  | R | U |  |  |  |  |  |  |  |  | D | C | RU |  |
| Online |  | C |  |  | U |  |  |  |  |  | U |  |  | D | C | RU |  |
| Wins |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C | RU |  |
| Losses |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C | RU |  |
| ScorePerMinute |  | C |  |  |  |  |  |  |  | RU |  |  | RU | D | C | RU |  |
| ActiveTile |  | C |  |  |  | C | R | U |  |  | U |  |  | D | C | RU |  |
| **Game** |  |  |  |  | R | C | R | U |  | U |  | R | U |  |  |  | R |
| Game ID |  |  |  |  | R | C | R |  |  |  |  | R |  |  |  |  | R |
| GameName |  |  |  |  | R | C | R |  |  |  |  | R |  |  |  |  |  |
| Game Status |  |  |  |  |  | C | R |  |  | U | U | R | U |  |  |  |  |
| **Session** |  |  |  |  | R | C | CR | U |  | R |  | R | R | D |  |  |  |
| Player ID |  |  |  |  | R | C | CR |  |  |  |  | R | R | D |  |  |  |
| Game ID |  |  |  |  | R | C | CR |  |  |  |  | R | R | D |  |  |  |
| BirdCount |  |  |  |  |  | C | CR | RU | R |  | R |  |  | D |  |  |  |
| Score |  |  |  |  |  | C | CR | U | RU | R | R |  | R | D |  |  |  |
| **Board** |  |  |  |  |  | C | R | RU |  |  |  |  |  |  |  |  |  |
| Board ID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Board Type |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| **Player Board** |  |  |  |  |  | C | CR |  |  |  | U |  |  | D |  |  |  |
| Player ID |  |  |  |  |  | C | U |  |  |  | U |  |  | D |  |  |  |
| Board ID |  |  |  |  |  | C | R |  |  |  |  |  |  | D |  |  |  |
| Player Number |  |  |  |  |  | C | U |  |  |  | U |  |  | D |  |  |  |
| **Game Board** |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Game ID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Board ID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Board Number |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| **Board Tile** |  |  |  |  |  | C | R | RU |  |  |  |  |  |  |  |  |  |
| Tile ID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Board ID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile Row |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile Column |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile Status |  |  |  |  |  | C | R | U |  |  |  |  |  |  |  |  |  |
| **TileType** |  |  |  |  |  | CR | R | RU |  |  |  |  |  |  |  |  |  |
| TileTypeID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile ID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| ItemID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| TypeDescription |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Inventory** |  |  |  |  |  | R | R | U |  |  |  |  |  | D |  |  |  |
| ItemplaceID |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
| PlayerID |  |  |  |  |  | R | R |  |  |  |  |  |  | D |  |  |  |
| ItemID |  |  |  |  |  | R | R | U |  |  |  |  |  |  |  |  |  |
| **Item** |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
| ItemID |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
| Type |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Chat** |  |  |  |  |  | C | R |  |  |  |  |  | R |  |  |  | U |
| PlayerID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  | U |
| GameID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  | U |
| Message |  |  |  |  |  | C |  |  |  |  |  |  |  |  |  |  | C |

## 6.1 C.R.U.D Analysis:

To develop the CRUD table above I used the entities from my ERD as the left hand column and the storyboards further up in this report as the actions along the top.

**Check For Username, Register account, Check for Password, Lock account and Successful Login:**

I decided to merge these events as it made the most sense to me, this event will be achieved by using a simple select where statement that looks for a specific username and password in the account table, if neither is found the user will be prompted to fill in extra information to register, if the username is found but the password isn’t/is incorrect then the user will be requested to try again and the login attempts column of the Account table will be incremented by one. The procedure will then check to see if the login attempts column is greater than or equal to 5 and if it is the account will become locked. If both username and password are correct the user will be logged in and taken to the home screen.

**Quick Join/Join Lobby:**

When a player joins a lobby the database will need to create new data as well as pull existing data from session and game tables. This will be more complex than just creating a game.

**Create Game:**

The database will need to populate the session and game tables with data so that someone wanting to join can pull the data from those tables.

**Player Move, Scoring:**

When the player moves the tile will be checked to see if there is a crow within a 1 tile space of it or if there is an item on the tile, if there is a crow present the crow will be “scared” and moved to a random tile on the opposing half, if there is an item it will be picked up. Score will be calculated at select intervals. When this is done the system will count the amount of tiles with a Crow status on the opponents half and add that many points to the points count, after this count, if a player has equal or greater to 100 points the game will be over.

**Game End:**

Upon game end the session will be concluded, the player with the most points or who reached 100 first will receive a you win message and the opponent will receive a you lose message. Both players Win, Loss and scoreperminute columns in the account table will be updated with new information.

**LogOff:**

Upon logout the players online status will be change to false and the ActiveTile column will be updated with with the tileid of the players last location.

**Admin:**

When the admin window is booted the database will retrieve the online players as well as session and game info. This will allow the admin to kill active games. The admin also has the power to create new records or edit records in the account table. This facilitates the addition of new players.

**Chat:**

When a player wants to send a chat it will update the chat table with their playerID and the message they want to type.

Milestone II

# 7. Procedures:

The gameplay architecture of Corvus revolves around the use of procedures with the intention for the final design to be imbedded in a C# repository to provide GUI Elements to the game. Below I will detail the development of several of the necessary procedures.

## 7.1 Player Registration:

To be able to play the game you require a user, this is one of the most integral procedures for the database. For the adduser procedure I wanted it to be a simple system that prompted upon the database not finding the requested username when login was attempted. Upon attempted login the procedure will attempt to select the username from the database. If the username is not present the procedure will insert data into the tblAccount table and return a “Welcome to Corvus” message.

## 7.2 Player Selection:

Player selection will revolve around the procedure to presenting a list of all online players through a select statement run on the online attribute of the tblAccount table. This field is updated upon login. The select statement will look for all Online attributes set to True and present the userID, username avgScore of each account that meets the criteria. This list will be displayed in descending order.

## 7.3 Creation and Confirmation for a Game:

When creating a new game there are several entities that will need to be populated. These include tblGame, tblSession, tblBoard, tblPlayerBoard, tblGameBoard, tblBoardTile.

Generating the initial game session will be a simple process conducted from the C# environment. Selected participants will be inserted into the game table. Once this is completed the next procedure will insert data into the board tables (Board, PlayerBoard, GameBoard, Boardtile) for each player.

## 7.4 Live Gameplay:

Once the game state has been created and the game has started the players will be able to move around their section of the board to begin scaring crows. When a player moves to a tile the surrounding tiles will be checked for crows. If a crow is present among these tiles it will be deleted and the same number of crows will be created on the opponents board.

When three moves have been made a procedure will run to count the number of crows on each players side and update the opposing players scoreboard by the crow count.

## 7.5 Game Administration Functions:

To maintain the environment of Corvus we require administration procedures. These procedures will be used to remove problematic players, update account details in case of password loss and end running games in case an error occurred.

To remove problematic players a simple delete account procedure will be used, searching the database for a specified username and when that username is found it will delete the associated account.

To update player info a procedure will be used to update or insert into the tblAccount. A select statement will be used to attempt to find the input username. If a username is not found the procedure will use an insert statement, if username is found then the procedure will use an update statement to add the new data to the tblAccount entry with the appropriate username.

To end a game a procedure will be used to remove all players from an active game by deleting the game entry with the supplied gameID.

# 8. ACID

ACID is a set of principles that help to insure that your database processes transactions reliably and without incident.

## Atomicity:

Atomicity ensures that all transactions execute correctly or fail but do not impact upon further transactions. The theme of atomicity is to make sure all transactions are individually wrapped and can’t be affected by previous transactions failing.

Atomicity is necessary for a database because it ensures that the database tables and entries will maintain integrity and all necessary functions will be carried out without interruption from the others. An example from our database would be when a player moves to a new tile, once the player moves there are several procedures that need to be run. Surrounding squares must be checked for crows, crows need to be added to opposing boards and scoreboards need to be updated. Without atomicity the game could become stuck if a crash happened before the opposing boards were updated with crows. Therefore no crows would appear on the opposing board once the game was resumed. This would then lead to a stalemate as nobody would be able to gain points.

## Consistency:

The Consistency principle revolves around maintaining the integrity of the database before and after transactions are run. Ensuring that invalid data is not inserted during transactions makes sure our database remains stable throughout any process that may be executed.

## Isolation:

The Isolation Principle states that every transaction committed in the database is self contained and not impactful on another transaction that may be running at the same time. Isolation is controlled through the use of Lock-based Concurrency Control or Multi-Versioned Concurrency Control.

## Durability:

The Durability Principle works to ensure that changes made with transactions remain a part of the database and that these additions are not lost. Durability revolves around adequate storage and backup options to ensure no data is lost.

# 9. Deviations from Milestone I Planning

## 9.1 Crud Table:

While developing my procedures I worked off of the CRUD table developed within the first milestone, looking out for any edits that need to be made. There were only a few edits I needed to make as I progressed through.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Entity/Attribute | **Check For Username** | **Register Account** | **Check For Correct Password** | **Lock Account** | **Successful Login** | **Start New Game** | **Quick Join/Join Lobby** | **Player Moves** | **Scoring** | **Game Ends** | **Player Logs Off** | **Open Admin Console** | **Admin Kill Running Game** | **Admin Delete Player** | **Admin Creates New Player** | **Admin Edits Existing Player Info** | **Player sends a Chat Message** |
| **Player** | R | C |  |  | R | R |  | R | R |  |  | R |  | D | C | RU | R |
| Player ID |  | C |  |  |  | R |  | R | R |  |  | R |  | D | C | RU | R |
| Username | R | CR | R |  | R |  |  |  |  |  |  | R |  | D | CR | RU | R |
| Password |  | C | R |  |  |  |  |  |  |  |  |  |  | D | C | RU |  |
| Email |  | CR |  |  |  |  |  |  |  |  |  |  |  | D | CR | RU |  |
| Locked User |  | C |  | U |  |  |  |  |  |  |  |  |  | D | C | RU |  |
| Admin |  | C |  |  | R |  |  |  |  |  |  |  |  | D | C | RU |  |
| Login Attempts |  | C |  | R | U |  |  |  |  |  |  |  |  | D | C | RU |  |
| Online |  | C |  |  | U |  |  |  |  |  | U |  |  | D | C | RU |  |
| Wins |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C | RU |  |
| Losses |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C | RU |  |
| ScorePerMinute |  | C |  |  |  |  |  |  |  | RU |  |  | RU | D | C | RU |  |
| ActiveTile |  | C |  |  |  | C | R | U |  |  | U |  |  | D | C | RU |  |
| **Game** |  |  |  |  | R | C | R | U |  | U |  | R | U |  |  |  | R |
| Game ID |  |  |  |  | R | C | R |  |  |  |  | R |  |  |  |  | R |
| GameName |  |  |  |  | R | C | R |  |  |  |  | R |  |  |  |  |  |
| Game Status |  |  |  |  |  | C | R |  |  | U | U | R | U |  |  |  |  |
| **Session** |  |  |  |  | R | C | CR | U |  | R |  | R | R | D |  |  |  |
| Player ID |  |  |  |  | R | C | CR |  |  |  |  | R | R | D |  |  |  |
| Game ID |  |  |  |  | R | C | CR |  |  |  |  | R | R | D |  |  |  |
| BirdCount |  |  |  |  |  | C | CR | RU | R |  | R |  |  | D |  |  |  |
| Score |  |  |  |  |  | C | CR | U | RU | R | R |  | R | D |  |  |  |
| **Board** |  |  |  |  |  | C | R | RU |  |  |  |  |  |  |  |  |  |
| Board ID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Board Type |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| **Player Board** |  |  |  |  |  | C | CR |  |  |  | U |  |  | D |  |  |  |
| Player ID |  |  |  |  |  | C | U |  |  |  | U |  |  | D |  |  |  |
| Board ID |  |  |  |  |  | C | R |  |  |  |  |  |  | D |  |  |  |
| Player Number |  |  |  |  |  | C | U |  |  |  | U |  |  | D |  |  |  |
| **Game Board** |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Game ID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Board ID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| Board Number |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  |  |
| **Board Tile** |  |  |  |  |  | C | R | RU |  |  |  |  |  |  |  |  |  |
| Tile ID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Board ID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile Row |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile Column |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile Status |  |  |  |  |  | C | R | U |  |  |  |  |  |  |  |  |  |
| **TileType** |  |  |  |  |  | CR | R | RU |  |  |  |  |  |  |  |  |  |
| TileTypeID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| Tile ID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| ItemID |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
| TypeDescription |  |  |  |  |  | C | R | R |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Inventory** |  |  |  |  |  | R | R | U |  |  |  |  |  | D |  |  |  |
| ItemplaceID |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
| PlayerID |  |  |  |  |  | R | R |  |  |  |  |  |  | D |  |  |  |
| ItemID |  |  |  |  |  | R | R | U |  |  |  |  |  |  |  |  |  |
| **Item** |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
| ItemID |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
| Type |  |  |  |  |  | R | R |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Chat** |  |  |  |  |  | C | R |  |  |  |  |  | R |  |  |  | U |
| PlayerID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  | U |
| GameID |  |  |  |  |  | C | R |  |  |  |  |  |  |  |  |  | U |
| Message |  |  |  |  |  | C |  |  |  |  |  |  |  |  |  |  | C |

## 9.2 ERD Re-Design:

When approaching the writing of the SQL I tried to maintain a minimalist approach, condensing it down wherever possible. As of writing this there have been no drastic edits to the ERD but this documentation will be updated whenever an edit occurs.